UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/849,813	05/04/2001	Shahaf Abileah	SVL920010035US1	1991

24852 7590 06/22/2004

INTERNATIONAL BUSINESS MACHINES CORP
IP LAW
555 BAILEY AVENUE , J46/G4
SAN JOSE, CA 95141

EXAMINER	
CAO, DIEM K	

ART UNIT	PAPER NUMBER
2126	7

DATE MAILED: 06/22/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/849,813	Applicant(s) ABILEAH ET AL.	
	Examiner Diem K Cao	Art Unit 2126	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *2-12-2002*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-49 are presented for examination.

2. The cross references related to the application cited in the specification must be updated (i.e. the relevant status, with PTO serial numbers or patent numbers where appropriate, on pages 1-2).

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-4, 14-16, 26-28, and 39-40 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-12 of copending Application No. 09/849,105. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "the transaction message formatter running on the application server" and "the metamodel data of the target transaction message

formatter including a message descriptor, logical page, password, segment, message field, device descriptor, device type, device division, device page and device field". It would have been obvious to one of ordinary skill in the art all the steps of the invention are the same in both applications, wherein COBOL and the transaction message formatter are different type of legacy applications provide functionality to the front end.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

5. Claims 1-39 and 41-49 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-47 of copending Application No. 09/849,190. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "the high level assembler running on the application server". It would have been obvious to one of ordinary skill in the art, the COBOL application and the high level assembler application are both legacy applications, and all the steps are identical in two applications.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

6. Claims 1-4, 11, 14-17, 24, 26-29, 37, 39-44 and 49 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over

claims 1-25 of copending Application No. 09/849,377. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "a language running on the application server". It would have been obvious to one of ordinary skill in the art COBOL is a programming language, and all the steps in the two applications are identical, therefore, one could apply any language instead of COBOL.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

7. Claims 1-49 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-49 of copending Application No. 09/849,563. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "PL/I running on the application server". It would have been obvious to one of ordinary skill in the art, the COBOL application and the PL/I application are both legacy applications, and all the steps are identical in two applications.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

8. Claims 1-4, 14-16, 26-28, 39 and 42 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-11 of copending Application No. 09/849,816. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "the transaction manager running on the application server". It would have been obvious to one of ordinary skill in the art all the steps of the invention are the same in both applications, wherein COBOL and the transaction manager are different type of legacy applications provide functionality to the front end.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

9. Claims 1-2, 4-7, 9-12, 14, 17-20, 22-25 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-18, of copending Application No. 10/310,343. Although the conflicting claims are not identical, they are not patentably distinct from each other because the only different between the instant application and the copending application is the instant application claims "COBOL running on the application server" and the copending application claims "a first target language running on the application server" and "said Type Descriptor class connector metamodels comprising of an Instance Type Descriptor Base class, a Type Descriptor Language Element class, and a Language Element Model inheriting from Type Description Language element class". It would have been obvious to one of ordinary skill in the art COBOL is a programming language, and it

could be replaced by another language because all the steps are identical in both applications and both share the same purpose.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 101

10. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

11. Claims 39-49 are rejected under 35 U.S.C. 101 because the disclosed invention is inoperative and therefore lacks utility. Claim 39 claims a product only without any function and use and the purpose of the product.

Claim Objections

12. Claims 1, 3, 16, 22, 26, and 28 are objected to because of the following informalities:

Claim 1 cites "theresponse" in step d should be "the response" and "arget language" in step (3) should be "target language".

Claims 3, 16, and 28 cite "an connector" should be "a connector".

Claim 26 cites "theapplication" should be "the application".

Claim 22 cites "The system of claim 23" should be "The system of claim 21".

Appropriate corrections are required.

Claim Rejections - 35 USC § 112

13. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

14. Claims 1-38 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 14, and 26 claim "COBOL running on the application server" or "COBOL language running on the application server" which is unclear. Examiner will interprets "COBOL application running on the application server" or "COBOL language application running on the server" for examining purpose.

Claim 1 cites in step (ii) "converting a response to the application request from the COBOL running on the application server as a source language to the first language of the first end user application as a target language", and sub-sequence steps (1), (2), and (3) cite "COBOL target language" are not consistent, i.e. convert from the COBOL source language to the COBOL target language. Examiner will interprets "COBOL target language" as "target language" in steps (1), (2) and (3) for examining purpose.

Claim 14 cites "to convert a response" in step (d) should be "to convert the response" because it refers to "a response" in step (d), and "wherein connector" in step (e) should be "wherein the connector" because it refers to "the connector" in steps (b), (c) and (d).

Claim 26 cites "the connector ... to receive the application request from the server" in step (c) should be "the connector ... to receive the response to the application request from the server", and "to convert a response" in step (c) should be "to convert the response".

Appropriate corrections are required.

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. Claims 1-4, 6-10, 14-17, 19-23, 26-29 and 31-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Devine et al (U.S. 2003/0191970 A1) in view of Yee et al (U.S. 6,738,975 B1).

17. **As to claim 1**, Devine teaches

- initiating the application request on the end user application in a first language with a first application program (The customer workstation ... and front-end services; page 4, section 0058 and client-tier software ... Java classes; page 4, left column, lines 1-5 and client message, the request; page 5, section 0068 and request from an application running on the customer's workstation; page 6, section 0074),
- transmitting the application request to the server (the messages created ... Web Servers 24; page 7, section 0095 and The DMZ ... the user section; page 5, section 0068) and

converting the application request from the first language of the first end user application to the legacy language application running on the application server (a translation process for translating a message into an underlying message; page 8, section 0099),

- processing the application request on the application server (other legacy or host platform ... at the client browser; page 5, section 0073),
- transmitting a response to the application request from the application server to the end user application, and converting the response from the legacy language to the first language of the first end user application (Data returned ... to client format ... response to the request; page 8, section 0099 and Any data returned from the application server is translated back to client format and returned over the Internet to the client workstation; page 6, section 0074), and
- wherein the end user application and the application server have at least one connector therebetween (legacy adapter; page 3, section 0054 and Fig. 1 and proxy; page 8, section 0099), and the steps of (i) converting the application request from the first language of the first end user application as a source language to the legacy language running on the application server as a target language (a translation process for translating a message into an underlying message; page 8, section 0099), and (ii) converting a response to the application request from the legacy language running on the application server as a source language to the first language of the first end user application as a target language (Any data returned from the application server is translated back to client format; page 6, section 0074) using the metamodels of respective source and target language (metadata; page 6, section 0081 and proxy specific data ... providing a service; page 8, section

0097), and converting the source language to the target language (a translation process for translating a message into an underlying message; page 8, section 0099 and Any data returned from the application server is translated back to client format; page 6, section 0074).

18. However, Devine does not teach COBOL language running on the application server, and populating the connector metamodels with metamodel data of each of the respective source and target languages. Devine teaches legacy applications in general, it would have been obvious to one of ordinary skill in the art the COBOL application could be run in the application server because COBOL application is legacy application. Yee teaches populating the connector metamodels with metamodel data of each of the respective source and target languages (The user identifies ... actual output message; col. 21, lines 1-11 and A transformer implements ... to the target objects; col. 21, lines 43-55).

19. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine and Yee because Yee provides a fast and simple integration of applications, legacy applications while reduce or substantially eliminate the need for the expensive custom coding that is traditionally required to integrate applications.

20. **As to claim 2**, Devine teaches the end user application is a web browser (The customer workstation ... browser based user interface; page 3, section 0056).

21. **As to claim 3**, Devine teaches the end user application is connected to the application server through a web server, and the web server comprises a connector (customer workstation, web server, application server, adapter programs; page 3, sections 0053-0055 and Fig. 1).

22. **As to claim 4**, Devine does not teach the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata. However, Devine teaches metadata is used (page 6, section 0081). Yee teaches connector (agent adapter) mediate differences interface protocols and data structures between applications to provide a uniform, normalized view of business event (col. 16, lines 62-65), message definition object identifies data of an enterprise application, and well-defined message format to describe the layout of the native data (col. 17, lines 49-54 and 48-61 and col. 19, lines 25-44), and mapping definition objects define how the system will transform system messages extract from one or more applications to messages needed by other applications (col. 17, lines 55-58 and col. 20, lines 53-59). Although Yee does not use the same terms as defined in the instant application, the metadata in the system of Yee covers the same as in the system of the application. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine and Yee because it will improve the system of Devine because the system of Yee provides an efficient and cost effective information sharing.

23. **As to claim 6**, Devine does not teach the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, output parameter

signatures, and return type. Yee teaches connector (agent adapter) mediates differences interface protocols and data structures between applications (col. 16, lines 62-65), and user can create a transformation to transform input data from the source application to the input format that expected by the target application (col. 21, lines 12-24). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine and Yee because it provides a method for communication between integrated applications.

24. **As to claim 7**, Devine does not teach the application metamodel metadata further includes language metamodel metadata. Yee teaches language metamodel metadata (The source adapter ... of their native data; col. 18, lines 48-61). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine and Yee because it provides a method to sharing data between multiple applications written in different languages.

25. **As to claim 8**, Devine does not explicitly teach the language metamodel metadata includes mappings between source and target language. Yee teaches the language metamodel metadata includes mappings between source and target language (The source adapter ... of their native data; col. 18, lines 48-61 and Mapping definition object ... by other applications; col. 17, lines 55-58).

26. **As to claim 9**, Devine teaches the source language is object oriented (Java class; page 4, section 0056). However, Devine does not teach the language metamodel metadata maps

encapsulated objects in to code and data. It would have been obvious to one of ordinary skill in the art, Java could pass object as an argument in the request, and the system of Devine translate the request into message of the format that is needed by the legacy application, which expect code and data, therefore, it would have been obvious the system of Devine would have to maps the object into code and data as expected by the legacy application.

27. **As to claim 10**, Devine does not teach the language metamodel metadata maps object inheritances into references and pointer. Devine teaches the source language is object oriented (Java classes; page 4, section 0056). It is well known in the art that Java class can inherent from parent class. It would have been obvious to one of ordinary skill in the art that in order to mapping from one language to another language, reference to parent class must be also mapped, and references and pointers are used in the object oriented to such purpose.

28. **As to claim 14**, it corresponds to the method claim of claim 1 above except it is a computer system claim.

29. **As to claims 15-17**, see rejections of claims 2-4 above.

30. **As to claims 19-23**, see rejections of claims 6-10 above.

31. **As to claim 26**, it corresponds to claim 14 above.

32. **As to claims 27-29**, see rejections of claims 2-4 above.

33. **As to claims 31-36**, see rejections of claims 6-10 above.

34. Claims 5, 11-13, 18, 24-25, 30 and 37-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Devine et al (U.S. 2003/0191970 A1) in view of Yee et al (U.S. 6,738,975 B1) further in view of Mellen-Garnett et al (U.S. 6,094,688).

35. **As to claim 5**, Devine does not teach the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data. Devine teaches message control information, security data and user data are used in the process of validating the request from the client before it could be served by the server application (the request is come from a valid user; page 5, section 0068, logon user, section 0070, the logical message format ... web server 24, page 7, section 0095 – page 8, section 0097). However, Devine does not teach transactional semantic, trace and debug information, pre-condition and post-condition resources. Mellen-Garnett teaches transactional semantic (Transaction service 248; col. 7, lines 46-62), trace and debug information (Error and exception service 260; col. 10, lines 23-40), pre-condition and post-condition resources (transaction system resource; col. 9, lines 57-67 and transaction is interrupt; col. 10, lines 28-40). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine

and Mellen-Garnett because it will improve the performance of Devine's system because transaction service provides consistency across applications in the system.

36. **As to claim 11**, Devine does not teach the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and relaxation constraints. Mellen-Garnett teaches the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and relaxation constraints (Data transformation service ... syntactic and semantic transformation of data; col. 7, lines 23-45 and a connector includes an API manipulator ... message transformer; col. 19, lines 6-14). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Devine and Mellen-Garnett because it provides a method for sharing information between applications written in different languages in the computing system.

37. **As to claim 12**, Devine does not teach the transaction is a rich transaction comprising a plurality of individual transactions, and further comprising processing the plurality of individual transactions on one end user application and a plurality of application servers. Mellen-Garnett teaches the transaction is a rich transaction comprising a plurality of individual transactions, and further comprising processing the plurality of individual transactions on one end user application and a plurality of application servers (transaction, sub-transaction; col. 8, lines 1-12 and individual connectors associated with the applications; col. 9, lines 1-17 and Fig. 1).

38. **As to claim 13**, Devine does not teach passing individual transactions among individual application servers. Mellen-Garnett teaches passing individual transactions among individual application servers (compensating transaction may e transferred to one or more connector; col. 9, lines 45-56 and 1-17).

39. **As to claims 18 and 30**, see rejection of claim 5 above.

40. **As to claims 24-25**, see rejections of claims 11-12 above.

41. **As to claims 37-38**, see rejection of claims 11-12 above.

42. Claims 39-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yee et al (U.S. 6,738,975 B1) in view of Mellen-Garnett et al (U.S. 6,094,688).

43. **As to claim 39**, Yee teaches connector (agent adapter) mediate differences interface protocols and data structures between applications to provide a uniform, normalized view of business event (col. 16, lines 62-65), message definition object identifies data of an enterprise application, and well-defined message format to describe the layout of the native data (col. 17, lines 49-54 and 48-61 and col. 19, lines 25-44), and mapping definition objects define how the system will transform system messages extract from one or more applications to messages needed by other applications (col. 17, lines 55-58 and col. 20, lines 53-59). Although Yee does not use the same terms as defined in the instant application, the metadata in the system of Yee

covers the same as in the system of the application. Yee further teaches a metamodel metadata repository of source and target language metamodel metadata (The repository service 135 ... metadata; col. 15, lines 55-58).

44. However, Yee does not explicitly teach invocation metamodel metadata and Cobol target language. Mellen-Garnett teaches invocation metamodel metadata (API manipulator; col. 19, lines 6-10). Yee teaches legacy applications in general, it would have been obvious to one of ordinary skill in the art the COBOL application could be run in the application server because COBOL application is legacy application.

45. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Yee and Mellen-Garnett because it improves the method for sharing information between applications written in different languages in the computing system.

46. **As to claim 40**, Yee teaches building connector stubs from the metamodel metadata (create agent-adapter; col. 26, lines 47-62 and generate customized intelligent agent-adapter; col. 15, lines 41-54).

47. **As to claim 41**, Yee teaches retrieving connector metamodel metadata of respective source and target languages from the metamodel metadata repository (The repository service ... metadata; col. 15, lines 55-58), populating the connector metamodels with metamodel data of

each of the respective source and target language from the repository (the user crate ... as output; col. 21, lines 1-11), invoking the retrieved, populated connector metamodels (When the transformer ... input messages; col. 21, lines 49-55), and converting the source language to the target language (transforms the input data; col. 21, lines 49-55 and col. 21, line 65 – col. 22, line 6).

48. **As to claim 42,** Yee teaches connector (agent adapter) mediate differences interface protocols and data structures between applications to provide a uniform, normalized view of business event (col. 16, lines 62-65), message definition object identifies data of an enterprise application, and well-defined message format to describe the layout of the native data (col. 17, lines 49-54 and 48-61 and col. 19, lines 25-44), and mapping definition objects define how the system will transform system messages extract from one or more applications to messages needed by other applications (col. 17, lines 55-58 and col. 20, lines 53-59). Although Yee does not use the same terms as defined in the instant application, the metadata in the system of Yee covers the same as in the system of the application.

49. **As to claim 43**, Yee teaches message control information (when the user define a message definition ... considered valid within the system; col. 19, lines 40-44), security data (passwords and user Ids; col. 20, lines 38-40), and user data (passwords and user Ids; col. 20, lines 38-40). However, Yee does not teach the invocation metamodel metadata is chosen from the group consisting of transactional semantics, trace and debug information, pre-condition and post-condition resources. Mellen-Garnett teaches transactional semantic (Transaction service

248; col. 7, lines 46-62), trace and debug information (Error and exception service 260; col. 10, lines 23-40), pre-condition and post-condition resources (transaction system resource; col. 9, lines 57-67 and transaction is interrupt; col. 10, lines 28-40). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Yee and Mellen-Garnett because it will improve the performance of Yee's system because transaction service provides consistency across applications in the system.

50. **As to claim 44**, Yee teaches connector (agent adapter) mediates differences interface protocols and data structures between applications (col. 16, lines 62-65), and user can create a transformation to transform input data from the source application to the input format that expected by the target application (col. 21, lines 12-24). Although Yee does not use the same terms as of instant application (input parameter signature, output parameter signature, and return type), one of ordinary skill in the art at the time the invention was made to know that the system of Yee provides the same functionality as claimed in the instant application.

51. **As to claim 45**, Yee teaches language metamodel metadata (The source adapter ... of their native data; col. 18, lines 48-61).

52. **As to claim 46**, Yee teaches the language metamodel metadata includes mappings between source and target language (The source adapter ... of their native data; col. 18, lines 48-61 and Mapping definition object ... by other applications; col. 17, lines 55-58).

53. **As to claim 47**, Yee does not teach the source language is object oriented, and the language metamodel metadata maps encapsulated objects in to code and data. Yee teaches all types of applications could be integrated into the system. It would have been obvious to one of ordinary skill in the art, the source language could be object oriented, and in object oriented programming, object could be passed as an argument in the request, and the system of Yee translate the request into message of the format that is needed by the other application, which expect code and data, therefore, it would have been obvious the system of Yee would have to maps the object into code and data as expected by the legacy application.

54. **As to claim 48**, Yee does not teach the language metamodel metadata maps object inheritances into references and pointer. Yee teaches all types of applications could be integrated into the system. It would have been obvious to one of ordinary skill in the art, the source language could be object oriented. It is well known in the art that in object oriented programming, class can inherent from parent class. It would have been obvious to one of ordinary skill in the art that in order to mapping from one language to another language, reference to parent class must be also mapped, and references and pointers are used in the object oriented to such purpose.

55. **As to claim 49**, Yee does not teach the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and relaxation constraints. Mellen-Garnett teaches the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and relaxation constraints (Data transformation

service ... syntactic and semantic transformation of data; col. 7, lines 23-45 and a connector includes an API manipulator ... message transformer; col. 19, lines 6-14). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Yee and Mellen-Garnett because it provides a method for sharing information between applications written in different languages in the computing system.

Conclusion

56. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Butts et al. (U.S. 6,233,542 B1) teaches "Server and terminal emulator for persistent connection to a legacy host system with response time monitoring".
- Mutschler, III (U.S. 6,253,366 B1) teaches "Method and system for generating a compact document type definition for data interchange among software tools".

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Diem K Cao whose telephone number is (703) 305-5220. The examiner can normally be reached on Monday - Thursday, 9:00AM - 5:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any response to this action should be mailed to:
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Diem Cao